Crypto Programming with GCrypt

What is GCrypt

- General purpose cryptographic library
- Developed as a separated module of GnuPG (Gnu Privacy Guard)
- Can be used independently of GnuPG
- Written in C
- Other languages can use it through a wrapper

What it Includes

- Basically all cryptographic building blocks
- Symmetric ciphers (AES, DES, Blowfish…)
- Hash algorithms (MD4, MD5, SHA-1…)
- MACs (HMAC, CMAC, GMAC)
- Public key algorithms (RSA, DSA…)

Installing Libgcrypt

- Also need Libgpg-error library
- Find them both at this link: https://www.gnupg.org/download/index.html and click download under the Tarball for each (can also download the signature and verify the files for practice/safety if you'd like)
- Extract the tar files into a folder as follows:

  ```
  sudo tar -xjvf <tar-file>
  ```

- Installing libgpg-error first: In the directory you extract libgpg-error to, run these commands:
  - `./configure --prefix=/usr --disable-static && make`
  - `sudo make install`
- Installing libgcrypt: In the directory you extract libgcrypt to, run these commands:
  - `./configure --prefix=/usr && make`
  - `sudo make install`

How to Use it

- #include <gcrypt.h>
- Function and type names all prefixed with gcry_ and symbols with GCRY_
- To ensure libgcrypt is found to be included, compile as follows:

  ```
  gcc -c foo.c `libgcrypt-config --cflags`
  gcc -o foo foo.o `libgcrypt-config --libs`
  gcc -o foo foo.c `libgcrypt-config --cflags --libs`
  ```

Initializing Gcrypt Library

- Start initialization:

```
gcry_check_version (GCRYPT_VERSION)
```

- If you want Secure Memory:

```
gcry_control (GCRYCTL_SUSPEND_SECMEM_WARN);
gcry_control (GCRYCTL_INIT_SECMEM, <secure_bytes>, 0);
gcry_control (GCRYCTL_RESUME_SECMEM_WARN);
```

- If not:

```
gcry_control (GCRYCTL_DISABLE_SECMEM, 0);
```

- Tell Gcrypt you’re finished initialization:

```
gcry_control (GCRYCTL_INITIALIZATION_FINISHED, 0);
```

Symmetric Crypto - Setup

- Constants used to define which algorithm: we use (DES, AES, Blowfish, etc.)

GCRY_CIPHER_<algo>

- Constants used to define mode of cipher (ECB, CFB, CBC, etc.)

GCRY_CIPHER_MODE_<mode>

- Creates cipher handle and places it at hd

gcry_error_t gcry_cipher_open (gcry_cipher_hd_t *hd, int algo, int mode, unsigned int flags)

- Sets key k on existing cipher handle h (needed for any encryption/decryption)

gcry_error_t gcry_cipher_setkey (gcry_cipher_hd_t h, const void *k, size_t l)

- Sets the initialization vector on existing cipher handle h (needed for all modes except ECB)

gcry_error_t gcry_cipher_setiv (gcry_cipher_hd_t h, const void *k, size_t l)

Symmetric Crypto – Actions and Teardown

– Encrypts in buffer to out buffer (or just encrypts out in place if in is NULL and inlen is 0)

```
gcry_error_t gcry_cipher_encrypt (gcry_cipher_hd_t h, unsigned char *out,
    size_t outsize, const unsigned char *in, size_t inlen)
```

– Decrypts in buffer to out buffer (or just decrypts out in place if in is NULL and inline is 0)

```
gcry_error_t gcry_cipher_decrypt (gcry_cipher_hd_t h, unsigned
    char *out, size_t outsize, const unsigned char *in, size_t inlen)
```

– Releases cipher handle and zeroes all sensitive information

```
void gcry_cipher_close (gcry_cipher_hd_t h)
```

Public Key Crypto

– **Encrypts data into r_ciph using the public key pkey**

gcry_error_t gcry_pk_encrypt (gcry_sexp_t *r_ciph, gcry_sexp_t data, gcry_sexp_t pkey)

– **Decrypts data into r_plan using the private key skey**

gcry_error_t gcry_pk_decrypt (gcry_sexp_t *r_plain, gcry_sexp_t data, gcry_sexp_t skey)

– **Creates a digital signature r_sig for data using the key skey**

gcry_error_t gcry_pk_sign (gcry_sexp_t *r_sig, gcry_sexp_t data, gcry_sexp_t skey)

– **Verifies that signature sig matches data using the public key pkey**

gcry_error_t gcry_pk_verify (gcry_sexp_t sig, gcry_sexp_t data, gcry_sexp_t pkey)

S-Expressions

- In the previous slide we saw a lot of variables of the type “gcry_sexp_t”
- These are a variation on LISP S-Expressions, see http://people.csail.mit.edu/rivest/sexp.html for more details
- gcry_error_t gcry_sexp_build (gcry_sexp_t *r_sexp, size_t *erroff, const char *format, ...) – Easiest way to build S-expressions. Builds expression from format string and stores it at r_sexp
- Use https://www.gnupg.org/documentation/manuals/gcrypt/Used-S_002dexpressions.html#Used-S_002dexpressions as a reference for the formats

Hashing

- Specify the hash algorithm to be used to generate the message digest

```
GCRY_MD_<hashing algorithm>
```

- Create message digest object and assign it to hd

```
gcry_error_t gcry_md_open (gcry_md_hd_t *hd, int algo, unsigned int flags)
```

- Add more hash algorithms to hd

```
gcry_error_t gcry_md_enable (gcry_md_hd_t h, int algo)
```

- Pass buffer into the handle h to update the digest value

```
void gcry_md_write (gcry_md_hd_t h, const void *buffer, size_t length)
```

- Pass the byte c into handle h to update the digest value

```
void gcry_md_putc (gcry_md_hd_t h, int c)
```

- Finalize the calculation and return the message digest

```
unsigned char * gcry_md_read (gcry_md_hd_t h, int algo)
```

- Close the hash context h and zero sensitive information. Can also copy and reset the hash context (to hash separate blocks from a certain state or the start state). See documentation for details

```
void gcry_md_close (gcry md hd t h)
```

Message Authentication Codes (MAC)

- Defines the MAC algorithm to use (e.g. GCRY_MAC_HMAC_SHA256)

GCRY_MAC_<MAC>_<hash algorithm>

- Create a MAC object at hd for algorithm algo

gcry_error_t gcry_mac_open (gcry_mac_hd_t *hd, int algo, unsigned int flags, gcry_ctx_t ctx)

- Set the key for the MAC

gcry_error_t gcry_mac_setkey (gcry_mac_hd_t h, const void *key, size_t keylen)

- Update the MAC value by passing buffer to the MAC object h

gcry_error_t gcry_mac_write (gcry_mac_hd_t h, const void *buffer, size_t length)

- Reads out the calculated MAC value into buffer

gcry_error_t gcry_mac_read (gcry_mac_hd_t h, void *buffer, size_t *length)

- Verify that a previously read MAC in buffer is the same as the MAC in h

gcry_error_t gcry_mac_verify (gcry_mac_hd_t h, void *buffer, size_t length)

- Closes the MAC object h and zeroes sensitive data

void gcry_mac_close (gcry_mac_hd_t h)

Other things

- Key Derivation
- “Truer” Random Numbers
- MPI Numbers (multi-precision integers for handling large numbers needed for crypto)
- Prime Numbers
- Shell tool for HMAC-SHA-256

An Example

- Code generally taken from http://cboard.cprogramming.com/c-programming/105743-how-decrypt-encrypt-using-libgcrypt-arc4.html#post937372

Activity

- Modify the previous example to use DES
- Use different cipher modes as well

Useful Links

- Home Page: http://www.gnu.org/software/libgcrypt/
- Download link: https://www.gnupg.org/download/index.html#libgcrypt
- Manual: https://www.gnupg.org/documentation/manuals/gcrypt/
- Good demo of public key generation and AES encryption: https://github.com/vedantk/gcrypt-example